INTEGRATION OF INFORMATION DISTRIBUTION SYSTEMS

RELATED APPLICATION

[0001] This application claims priority under 35 U.S.C. § 119 based on U.S. Provisional Application No. 60/488,798 filed July 22, 2003, the disclosure of which is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

A. Field of the Invention

[0002] The present invention relates generally to operations support systems (OSSs), and more particularly to switched service software systems.

B. Description of Related Art

[0003] In today's computerized world, companies may use sophisticated software systems to coordinate and manage many aspects of their business. The software systems are often highly customized to the particular needs of the company and can be a significant asset. When two competitive companies within the same business sector merge into one bigger company, reusing existing resources in each company becomes a top priority systems integration task. In particular, it is often desirable to integrate the software systems of the two companies into a single system. Such an integration can be difficult to perform, but once implemented, can provide large potential cost savings and efficiency improvements.

[0004] As an example of the above, consider the merger of two telecommunications companies. One of the valuable reusable resources for the newly consolidated company is the software systems from each company.

These systematically reusable software systems keep the company business going, supports customer services, and create revenue. For the new merged company, consolidating software systems with the same functionality, such as provisioning systems and inventory systems, can provide significant benefits.

[0005] Integrating software systems, although potentially beneficial, also entails risk. The risk is because the software systems may have different data representations, business processes, business rules, and system implementations, as well as different hardware platforms that support these software systems. These factors make efficiently integrating these software systems without introducing errors or faults a difficult proposition.

[0006] Thus, there is a need in the art for techniques for integrating information distribution systems that reduce the cost and/or risk involved in such integrations.

SUMMARY OF THE INVENTION

[0007] One aspect of the invention is directed to a method of integrating software systems. The method includes identifying a scope of the integration based on a multi-level top-down approach and identifying faults in business rules that define software in the scope of the integration by applying generic depth-first search (DFS)-based techniques to the business rules. The method further includes modifying the business rules based on the identified faults.

[0008] A second aspect of the invention is directed to a system for integrating information distribution systems. The system includes means for assisting a user to identify a scope of the integration using a multi-level top-down approach, the identified scope including a set of business processes that are to be integrated and a set of business rules that define the business processes. The system further includes a fault detection component configured to identify faults in the business rules by applying generic depth-first search based techniques to the business rules.

[0009] Yet another aspect of the invention is directed to a method of integrating information distribution systems of merging entities. The method includes identifying top-level software systems that are to be integrated, identifying business processes in the top-level software systems, and comparing the identified business processes to determine business processes that are related enough to be candidates for integration. The method further includes identifying business rules that define the business processes and identifying faults in the business rules by applying generic depth-first search based techniques to the business rules.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0011] Fig. 1 is a diagram of an exemplary computing device that may be used to implement aspects of the invention;

[0012] Fig. 2 is a diagram conceptually illustrating a five-level top-down integration framework that may be used consistent with aspects of the invention for integrating information distribution systems;

[0013] Fig. 3 is a diagram illustrating exemplary relationships between the component levels shown in Fig. 2;

[0014] Fig. 4 is a flow chart illustrating operations for integrating two information distribution systems;

[0015] Fig. 5 is a block diagram illustrating an exemplary set of identified software systems;

[0016] Fig. 6 is a diagram illustrating an exemplary set of rules represented using a Transition-Directed-Graph (TDG) representation;

[0017] Fig. 7 is a diagram graphically illustrating six exemplary categories of chained-inference faults; and

[0018] Fig. 8 is a block diagram graphically illustrating an operation shown in Fig. 4.

DETAILED DESCRIPTION

[0019] The following detailed description of the invention refers to the accompanying drawings. The detailed description does not limit the invention.

[0020] As described herein, an integration component assists in the integration of information distribution systems (IDSs). The integration component may help to identify the scope of the systems that are to be integrated, such as the software systems, business processes, business rules, interface functions, and data that are to be integrated. The integration component may then check the identified systems to identify potential faults in the integrated systems. This check can be performed by representing the business rules as a Transition-Directed-Graph (TDG) and then processing the TDG using generic depth-first search (DFS)-based algorithms.

SYSTEM OVERVIEW

[0021] Many of the techniques described herein are implemented on or with assistance of a computing device. Fig. 1 is a diagram of an exemplary computing device 100. Computing device 100 may include a bus 110, a processor 120, a main memory 130, a read only memory (ROM) 140, a storage device 150, input device(s) 160, one or more an output devices 170, and a communication interface 180. Bus 110 may include a set of conductors that permit communication among the components of computing device 100.

[0022] Processor 120 may include a conventional processor or microprocessor that interprets and executes instructions. Main memory 130 may include a random access memory (RAM) or another type of dynamic storage device that stores information and instructions for execution by processor 120. ROM 140 may include a conventional ROM device or another type of static storage device that stores static information and instructions for use by processor 120. Storage device 150 may include a magnetic and/or optical recording medium and its corresponding drive.

[0023] Input device(s) 160 may include conventional mechanisms that permit a user to input information to computing device 100, such as a keyboard, a mouse, a pen, voice recognition and/or biometric mechanisms, etc. Output devices 170 may include conventional mechanisms that output information to the user, including a display, a printer, a speaker, etc. Communication interface 180 may include any transceiver-like mechanism that enables computing device 100 to communicate with other devices and/or systems.

[0024] As will be described in detail below, computing device 100, consistent with principles of the invention, may implement an integration component 105. Integration component 105 may be stored in a computer-readable medium, such as memory 130. A computer-readable medium may be defined as a physical or logical memory device and/or carrier wave.

[0025] The software instructions defining integration component 105 may be read into memory 130 from another computer-readable medium, such as data storage device 150, or from another device via communication interface 180. The software instructions contained in memory 130 may cause processor 120 to perform processes that will be described later. Alternatively, hardwired circuitry or other logic may be used in place of or in combination with software instructions to implement processes consistent with the present invention. Thus, implementations consistent with the principles of the invention are not limited to any specific combination of hardware circuitry and software.

INTEGRATION FRAMEWORK

[0026] Fig. 2 is a diagram conceptually illustrating a five-level top-down integration framework that may be used consistent with aspects of the invention for integrating information distribution systems (IDSs). In general, software requirements for all processes that are to be integrated within the five levels are identified, to thus define the scope of the integration. This identification may be performed manually by an operator and/or with the assistance of integration component 105.

[0027] Software system level 201 may include the high-level software systems used by the two integrating parties. For example, telecommunication companies may have separate provisioning and billing software systems. These two systems would be identified as a separate system at the software system level.

[0028] Business process level 202 includes business processes, where a business process may be defined as a sequence of transitions, driven by events or data, which can be viewed as a set of chain-inference rules. In other words, a business process can be a collection of related, structured activities—a chain of events—that produce a specific business result. The result may be the creation of a product or service, or an intermediate component which contributes to the creation and delivery of products or services, either directly or indirectly. Examples of business processes may include a certain sales process, invoicing process, or marketing process.

[0029] Business processes in business process level 202 may each be defined by a set of business rules at business rule level 203. A set of business rules are employed to implement a business process. In other words, a business rule may be defined as a transition in a business process. Once a business process is identified, the business rules associated with the process can be identified and defined.

[0030] Interface function level 204 includes the interface functions that define communications between business rules. More particularly, the communications between two business rules may be defined as being performed through one or more interfaces. The interfaces include functions designed to effectuate the interfaces. The functions are employed to ensure a defined business rule is not broken during the information distribution. Once business rules are defined, functions can be documented into the integration requirements.

[0031] Data level 205 defines the data used by levels 201-204. Data integrity refers to the consistency of information stored within different information distribution systems. Data transfer across IDS/system boundaries should remain consistent in format and representation.

[0032] Fig. 3 is a diagram illustrating exemplary relationships between the component levels 201-205 in additional detail. System software 301 may include one or more business processes 302. As mentioned, a business process 302 can be defined as a sequence of transitions, where each transition may be represented by a business rule 303. Functions 304 implement business rules 303. Functions 304 may communicate with other functions through an interface 305. Functions 304 of business rules 303 may operate on data 306.

SYSTEM OPERATION

[0033] Fig. 4 is a flow chart illustrating operations for integrating two IDSs. The integration process may include identifying the different logical levels 201-205 in the two IDSs.

[0034] The operations for integrating IDSs may begin by identifying top level software systems (i.e., software systems level 201) that should be integrated (act 401). For two merging Telecommunications companies, for example, each may have software systems dedicated to communication provisioning. These provisioning systems may handle all aspects associated with identifying and activating communication services to customers.

[0035] As an illustrative example, Fig. 5 is a block diagram that depicts an exemplary set of software systems that may have been identified in act 401. As shown, the identified software systems include a provisioning system 501 from company “A,” a provisioning system 502 from company “B,” and a long distance system 503 from company “A.” Provisioning systems 501 and 502 are likely to be identified as performing many common functions. Systems 501 and 502 may thus become targets for integration.

[0036] Systems targeted for integration may then be analyzed at the business process level (act 402). One or more business processes may be identified in the software systems. A number of business processes are shown in Fig. 5 for provisioning systems 501 and 502. The business processes for provisioning system 501 are labeled as business processes 511-1 thru 511-N (collectively referred to as business processes 511). The business processes for provisioning system 502 are labeled as business processes 512-1 thru 512-M (collectively referred to as business processes 512). Various ones of the business processes, such as business process 511-1 and business process 512-2, may then be identified as processes that should be integrated (act 403).

Business processes that are to be integrated may be, for example, business processes that perform substantially the same or similar functions or processes that are somehow related and for which benefits could be obtained by integrating the processes. Thus, in general, business processes 511 and 512 are "compared" to locate related pairs (or potentially more than two) of business processes. For example, business processes 511-1 and 512-2 may both be processes relating to setting-up a virtual private network (VPN) connection. These two processes would be determined to be targeted for integration.

[0037] As previously mentioned, business processes may be implemented through sets of business rules. Business rules are defined for each of the identified business processes (act 404). Business rules may operate on data (e.g., customer information) exchanged through interfaces. The data and interfaces used to exchange data represent the data level 205 and interface function level 204 of an IDS. Accordingly, for each business rule, the type of data used by the rule and the interfaces through which the business rules interact may also be identified (act 405).

[0038] For each set of business processes that are to be integrated, it may be generally desirable that the integrated set of rules associated with the processes are consistent and do not interact to produce errors or other undesirable results. Potential faults in the business rules may, thus, be identified (act 406). In an exemplary aspect of the invention, an integrated set of rules may be represented using a Transition-Directed Graph (TDG) representation which may be further processed by integration component 105 using generic depth-first search (DFS)-based algorithms to identify potential faults in the rules (act 406). Based on the identified faults, the rules may be modified (act 407). The potential faults that are identified for the business rules may be based on inconsistency, contradiction, circularity, subsumption, redundancy, and incompleteness. To assist in the understanding of the invention, each of these faults will be described below. Additionally, the TDG representation and the DFS-based algorithms are generally known in the art, and are described in, for example, Y. Hwang, "Detecting Faults in Chained-Inference Rules in Information Distribution Systems," doctoral dissertation, George Mason University, Fairfax, VA, Dept. Computer Science, 1997.

[0039] Fig. 6 is a diagram illustrating an exemplary set of rules represented using a TDG representation. The set of rules for the TDG may represent a business process.

[0040] In Fig. 6, business rules are represented by lines and vertices. More specifically, the results of interface functions 304 are represented by vertices, such as vertex 601. Lines, such as lines 602, represent particular functions 304. A business rule may be initiated based on the occurrence of a predicate, labeled as predicates A, B, C, and D in Fig. 6.

[0041] Using the TDG representation, two or more predicates in the Right Hand Side (RHS) of one business rule are allowed. For instance, as depicted in Fig. 6, predicate A is involved in three business rules—r1, r2, and r4.

[0042] Based on the TDG representation, DFS-based algorithms can be used to identify six categories of chained-inference faults. Fig. 7 is a diagram graphically illustrating the six categories of chained-interference faults. The six categories are inconsistency 701, circularity 702, subsumption 703, contradiction 704, redundancy 705, and incompleteness 706 and 707. Each category depicts a pattern that describes a fault. Once one of fault patterns 701-707 is found, the computational complexity to locate the rules involved in the pattern is on the order of O(*n*), where *n* is the number of vertices in a given TDG.

[0043] Pattern 701, inconsistency, can be defined as a path that exists with a starting vertex that is conflicting with an ending vertex. In pattern 701, starting vertex 710 conflicts with ending vertex 711, as vertex 710 represents a predicate A and ending vertex 711 represents a predicate "not A." Pattern 702, circularity, can be defined as a path with an identical starting and ending vertex (i.e., a cycle). In pattern 702, starting vertex 720 is also the ending vertex. Pattern 705, redundancy, can be defined as a path in which a starting vertex reaches the same ending vertex along two or more paths. In pattern 705, starting vertex 750 and ending vertex 751 can both be reached by different paths 752 and 753. Pattern 704, contradiction, can be defined as a pattern that includes two paths with an identical starting vertex and contradicting ending vertices. In pattern 704, two paths begin from starting vertex 740 (predicate A) that end in contradicting predicates (predicate D and not D) at vertices 741 and 742. Pattern 703, subsumption, can be defined as a type of redundancy pattern 705, in which one of the redundant paths is a single rule. In pattern 703, paths 731 and 732 both lead to the ending vertex. Path 732 is a single rule. Patterns 706 and 707 illustrate incompleteness faults. An incompleteness fault can be defined as an internal vertex that cannot be reached from any one of the input vertices, or an internal vertex that cannot reach any of the output vertices.

[0044] The DFS-based techniques can be used to detect faults at both local and global levels at the same time. Possible criteria used to detect faults at both local and global levels will now be generally described below.

[0045] *Inconsistency.* During application of the DFS-based algorithm, if one vertex is active and is revisited, then pattern 701 may be considered to be found. A vertex is considered to be active if it has been visited and there exists one or more adjacent vertices that have not yet been visited.

[0046] *Contradiction.* During application of the DFS-based algorithm, a contradiction pattern is found when there exists an exclusive pair of vertices, such as vertices 741 and 742, and one distinguished vertex, such as vertex 740, and a path exists from vertex 740 to vertex 741 and a second path exists from vertex 740 to vertex 742.

[0047] *Circularity.* During application of the DFS-based algorithm, a contradiction pattern is found when a vertex is active and is revisited.

[0048] *Redundancy.* During application of the DFS-based algorithm, redundancy pattern 705 is found when a vertex is inactive and is revisited. A vertex is inactive if it has been visited and all its adjacent vertices have also been visited.

[0049] *Subsumption. S*ubsumption pattern 703 is a specific type of redundancy fault in which one of the paths is a single rule.

[0050] *Incompleteness*. After application of the DFS-based algorithm, incompleteness patterns 706 or 707 are found based on an unvisited vertex. Also, during application of the DFS-based algorithm, incompleteness patterns 706 or 707 are found if there exists an exclusive pair of vertices such that one vertex in the pair can reach the other, or vice versa.

[0051] Fig. 8 is a block diagram graphically illustrating the operations of act 406 (Fig. 4). As shown, a set of integrated rules 801 is input to fault detection component 802. Based on the TDG representation and DFS-based algorithms, fault detection component 802 generates a list of possible faults in rules 801. Based on the identified faults, defected rules may be modified and re-input to fault detection component 802 (act 407). In this manner, rules that define business processes (which in-turn define software systems) that are to be integrated can be efficiently checked and modified to remove faults before the rules are physically implemented in software code.

CONCLUSION

[0052] Techniques were described above that can be used to integrate two or more Information Distribution Systems. Within these techniques, a five-level based approach is used to identify each component that is to be integrated. The five-level approach may first start with identifying related software systems that are to be integrated. Sets of business processes within the identified software systems may then be established. Based on the established business processes, sets of business rules can be identified. The interface functions and data that are associated with the business rules are used to ensure the business rules are not broken and that data integrity between each integrated system is held.

[0053] The TDG can be used to represent each component to be integrated during the system integration. Based on the TDG representation, six different types of faults within the integrated business rules can be verified using DFS-based algorithms. Thus, using the TDG and DFS-based algorithms, the business rules and interfaces of the business process pairs can be automatically analyzed to find faults (conflicts). Detected faults can be resolved manually or potentially using automated techniques.

[0054] It will be apparent to one of ordinary skill in the art that aspects of the invention, as described above, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement aspects consistent with the present invention is not limiting of the present invention. Thus, the operation and behavior of the aspects were described without reference to the specific software code -- it being understood that a person of ordinary skill in the art would be able to design software and control hardware to implement the aspects based on the description herein.

[0055] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, although many of the operations described above were described in a particular order, many of the operations are amenable to being performed simultaneously or in different orders to still achieve the same or equivalent results.

[0056] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article “a” is intended to include one or more items. Where only one item is intended, the term “one” or similar language is used.